Advanced Analysis of Algorithms

K. Subramani, LCSEE, West Virginia University

November 12, 2013

1. Theory of **NP-completeness**.
2. What is a problem? A question to be answered over several parameters with unspecified values and a constraint on the type of solution that is desired.
3. Instance of a problem - Instantiating the parameters of the problem.
4. Size of input. Length of input string.
5. An algorithm - Step by step decision procedure that takes as input an instance and returns the correct answer.
6. TSP problem example.
7. Time complexity function.
8. Efficient algorithms and intractable problems. Exponential time worst-case algorithms may actually work well in practice. Simplex.
9. Problem classification I: Decision, function, search, optimization.
10. Every decision problem Π consists of a set of instances D_Π, Y_Π and N_Π.
11. The motive behind this class. The Garey-Johnson story.
12. Problem classification II: Tractable, Intractable, evidence of intractability.
13. Reasons for intractability $\rightarrow$ definitional, undecidable, Presburger.
14. Problem as a formal language. Alphabet, Strings, language, problem and encodings. Binary encodings. Everything can be transformed based on encoding. Give graph shortest paths, example.
15. Problem is membership question. Use even numbers and prime numbers as languages. Membership question is interesting when language is infinite.
16. Associated with each decision problem Π is the language L_Π, which is the set of strings in the encoding, such that $x \in Y_\Pi$.
17. Algorithm as decider. What does it mean for an algorithm to decide L. Associated with an algorithm $\mathcal{A}$ is its language $L_\mathcal{A}$.
18. Deterministic algorithms and the class **P**.
19. Non-deterministic algorithms and the class **NP**. Guess and check algorithm. Easy verifiability. The tree of computations method to illustrate nondeterminism. You count the depth of the tree and not the total number of computations.